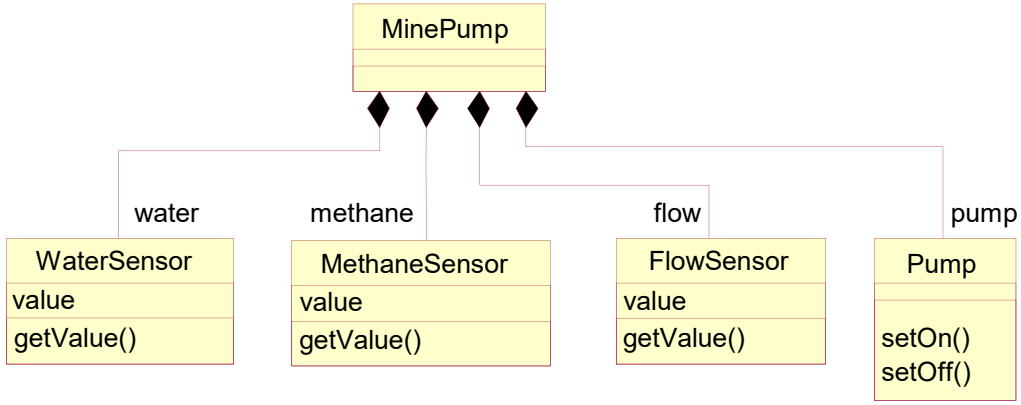
Design of Real-Time Software Workshop

Part 3: Workshop Solutions “Mine Pump”

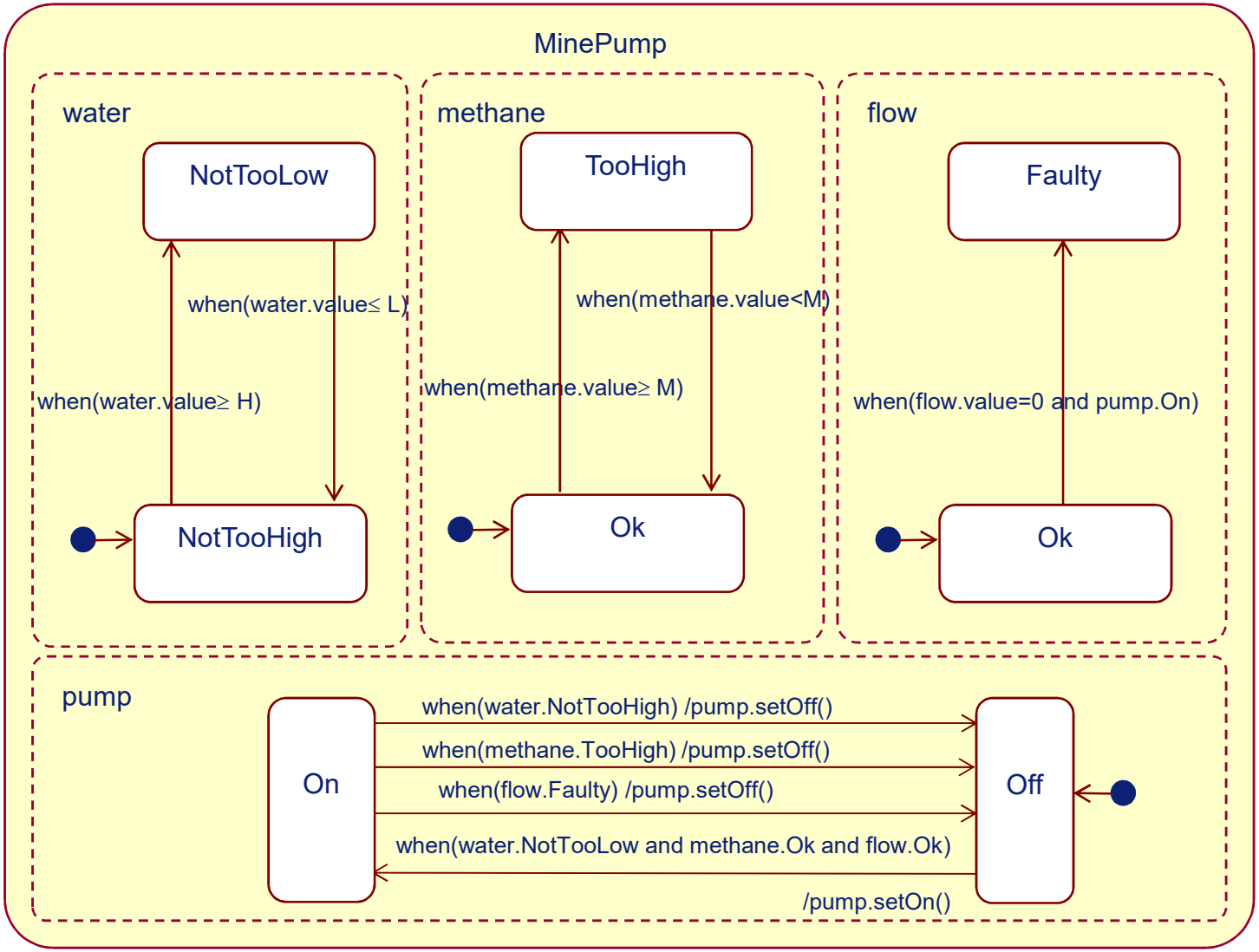
Onno van Roosmalen
O.S.vanRoosmalen@chello.nl

Version March 2010

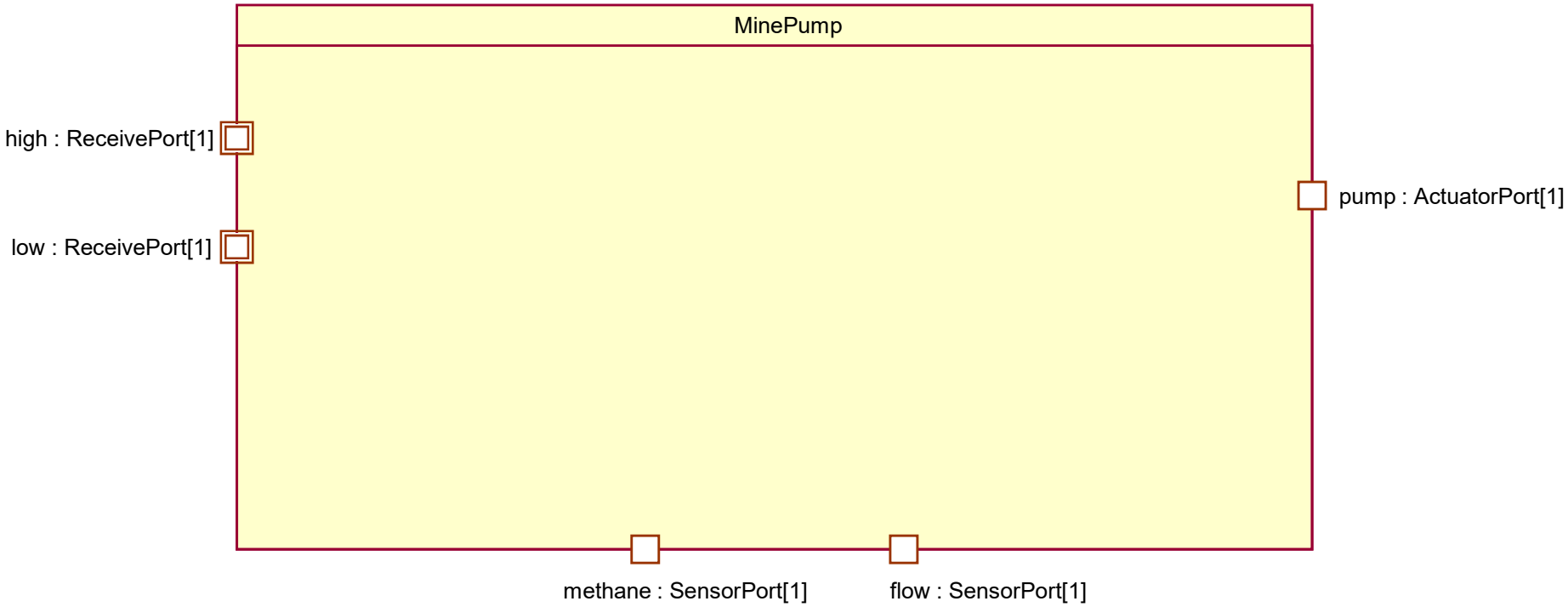
Analysis Class Diagram



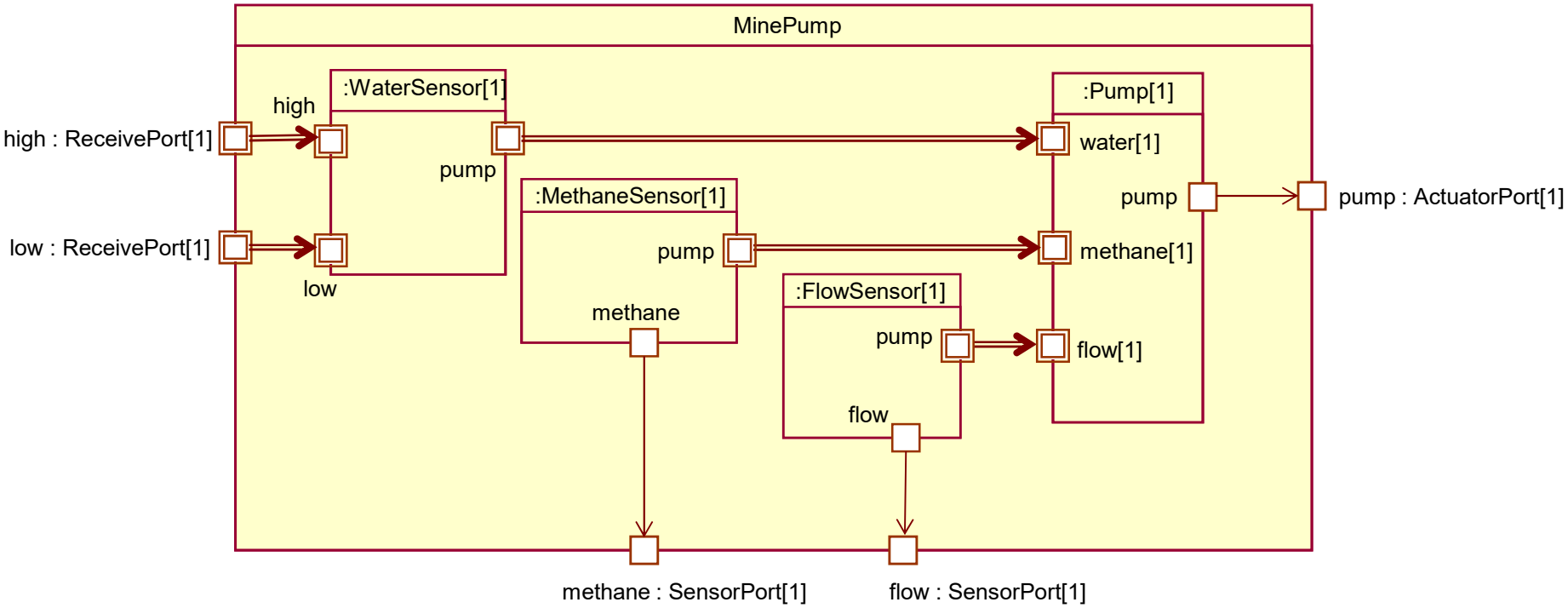
Analysis State Diagram



Design: Composite Structure with Ports

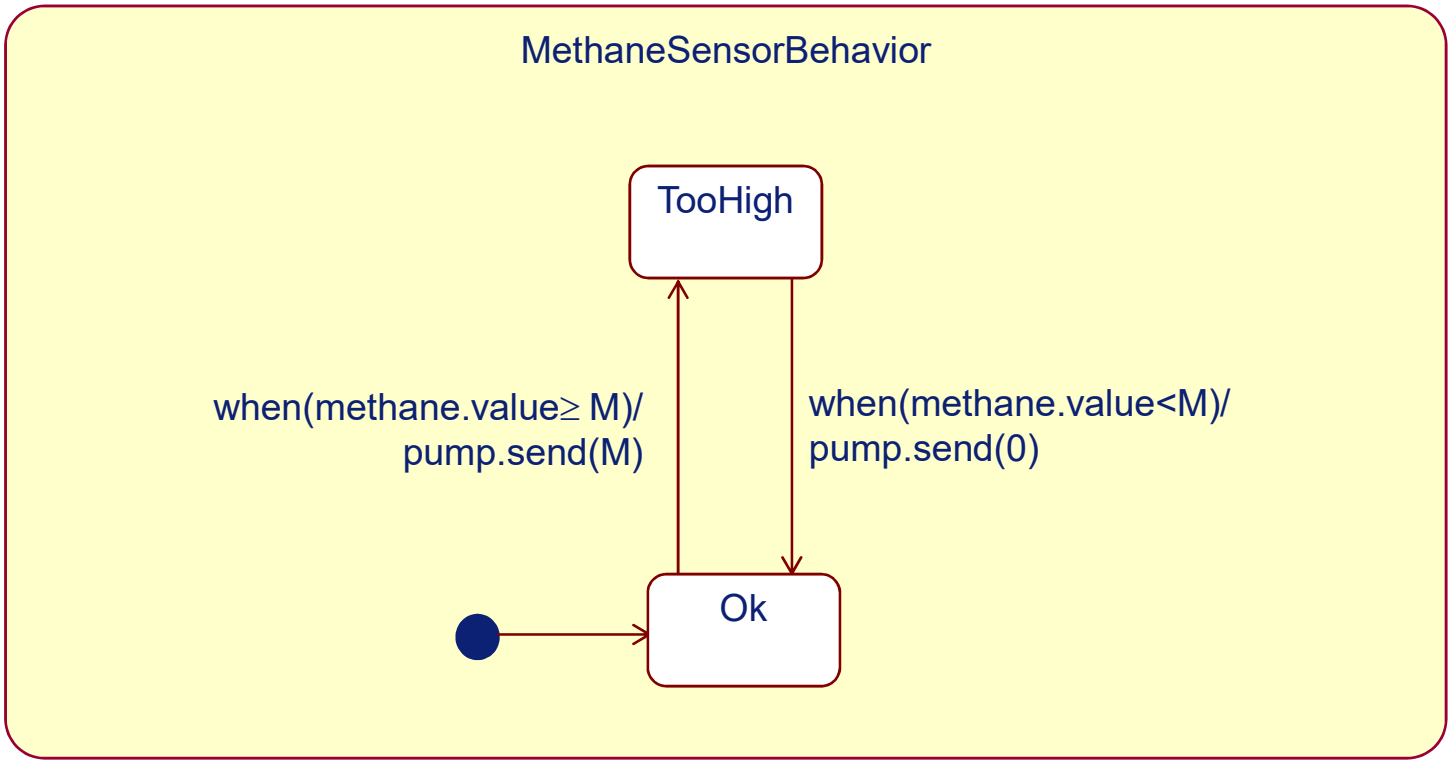


Refinement Step



Methane Sensor Behavior

$$\forall t_1, t_2 : (\text{methane} \geq M) \textbf{ during } [t_1, t_2] \rightarrow \text{pump} = \mathit{off} \textbf{ during } [t_1 + dc, t_2]$$



Language Mapping

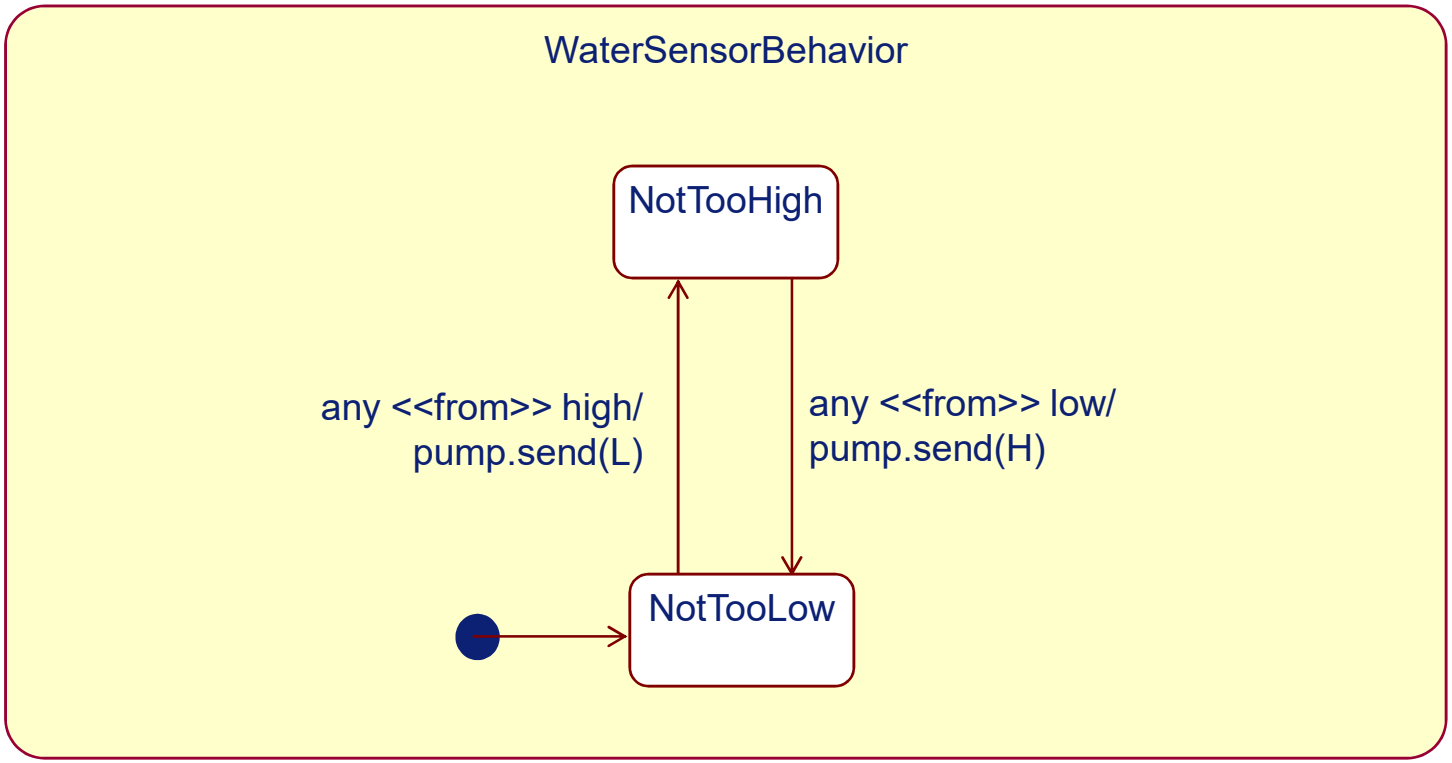
```
class MethaneSensor extends Thread {
        …
        public void run() {
                state = Ok;
                while (true) {
                        Value x = methane.getValue();
                        if ((x >= M) && (state==Ok)) {
                                state = TooHigh;
                                pump.send(M)
                        } else if (x < M && (state==TooHigh)) {
                                state = Ok;
                                pump.send(0);
                        }

                }
        }
}
```

Water Sensor behavior

$$\forall t_1, t_2 : (\text{water} \geq H \wedge safe)\ \textbf{during}\ [t_1, t_2] \rightarrow \text{pump} = on\ \textbf{during}\ [t_1 + dh, t_2]$$

$$\forall t_1, t_2 : (\text{water} \leq L)\ \textbf{during}\ [t_1, t_2] \rightarrow \text{pump} = off\ \textbf{during}\ [t_1 + dl, t_2]$$



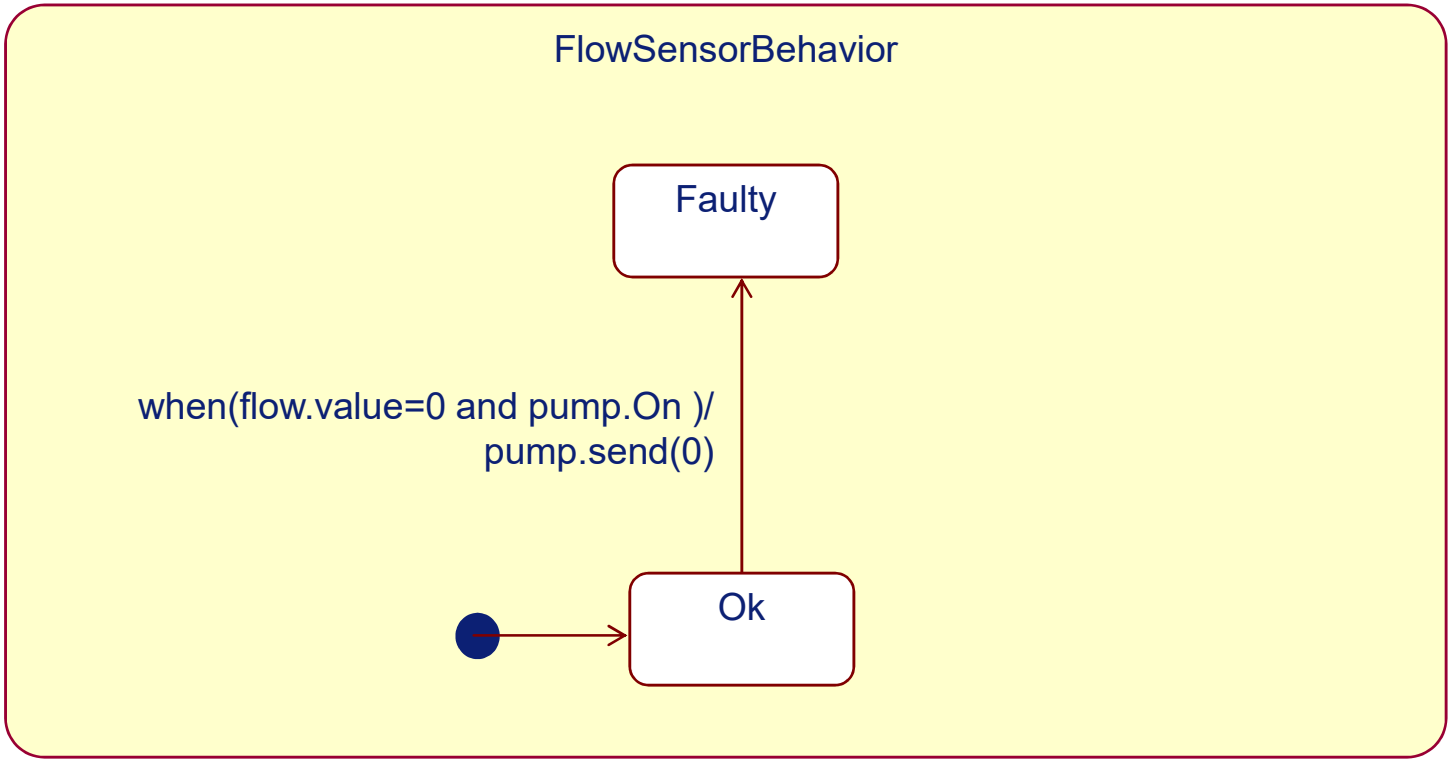
Language Mapping

```
class WaterSensor extends Thread {
        …
        public void run() {
                while (true) {
                        high.receive();
                        pump.send(H)
                        low.flush();
                        low.receive();
                        pump.send(L);
                        high.flush();
                }
        }
}
```

Flow Sensor Behavior

$$faulty \ \textbf{at} \ t \equiv \exists t_1 < t : (\text{flow} < min \wedge \text{pump} = on) \ \textbf{at} \ t_1$$

$$\forall t_1, t_2 : (faulty \wedge \text{pump} = on) \ \textbf{during} \ [t_1, t_2] \rightarrow \text{pump} = off \ \textbf{during} \ [t_1 + \delta f, t_2]$$



Naive Mapping FlowSensor

- The flow and pump status must hold simultaneously:

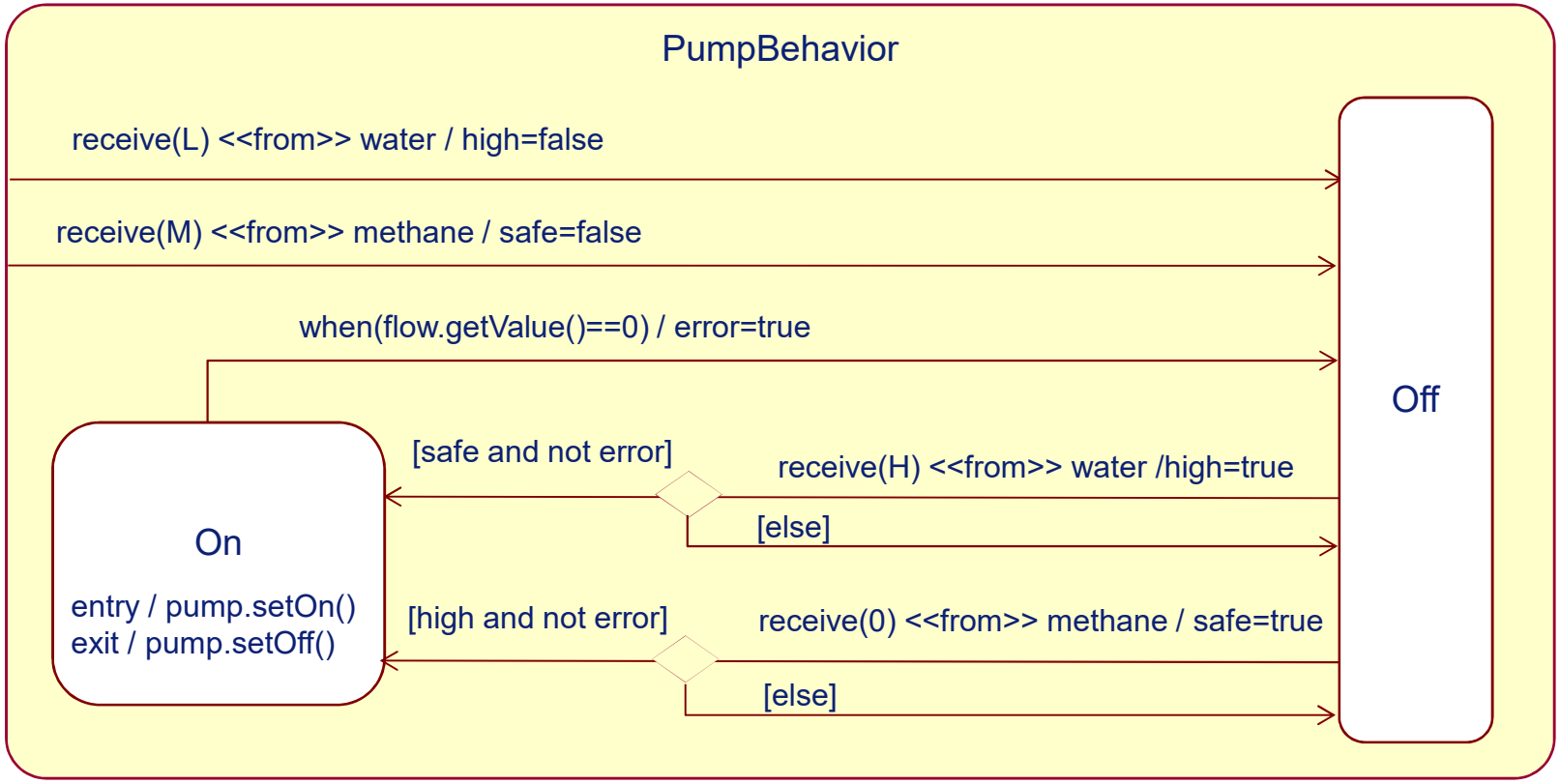
```
class FlowSensor extends Thread {
        …
        public void run() {
                state = Ok;
                while (Ok) {
                        if (pump.state==On) {
                                Value x = flow.getValue();
                                if ((x =0) {
                                        state = Faulty;
                                        pump.send(0);
                                }
                        }

                }
        }
}
```

Pump Process

- The Pump process must handle receptions from various ports
  - methane
  - flow
  - water
- It can not be occupied in an activity checking flow because it must receive signals
- It can not be blocking on reception of signals because it must check flow (when pump is on)
- We must also take care that pump state does no change during flow measurement

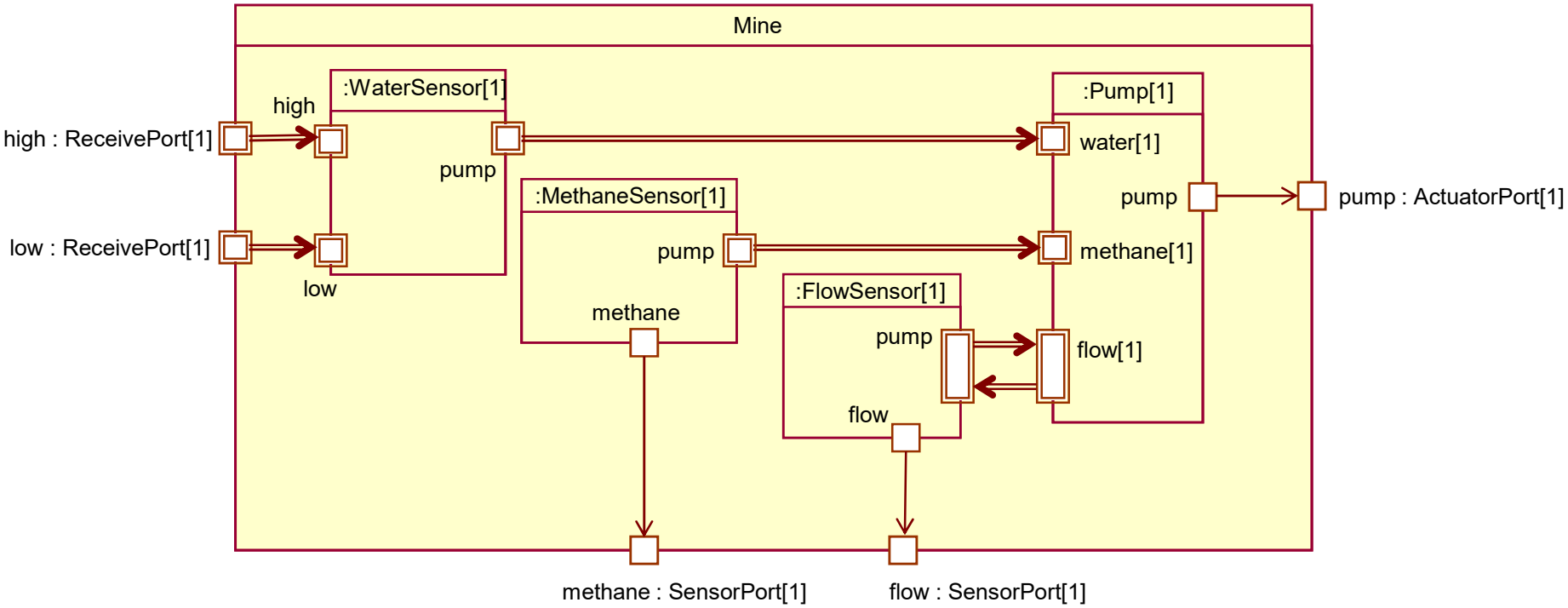
Pump Behavior?



Language Mapping

```
class Pump extends Thread {
        …
        public void run() {
                while (not error) {
                        port=receiveAll();
                        if (port==water) {
                                high=(port.receive()==H);
                        } else if(port==methane) {
                                safe=(port.receive()==M);
                        } else if(?) {
                                if (state==On) error=(flow.getValue==0);
                        }
                        if(state==Off && safe && high && !error ) {
                                state=On; pump.setOn();
                        } else if(state==On && (!safe || !high || error)) {
                                state=Off; pump.setOff();
                        }
                }
        }
}
```

Refinement Step (2)



Argument: …

```
class Pump extends Thread {
        …

        public void run() {
                error = false;
                while (!error) {
                        if (state==On) {
                                Value x = flow.getValue();
                                if ((x =0) {
                                        error = true;
                                        pump.setOff()
                                }
                        }

                }
        }
}
```

… Sequential Cohesion

```
class Pump extends Thread {
        …

        public void run() {
                error = false;
                while (!error) {
                        if (state==On) {
                                flow.send(); flow.receive(x);
                                if ((x =0) {
                                        error = true;
                                        pump.setOff()
                                }
                        }

                }
        }
}
```

FlowSensor

```
class FlowSensor extends Thread {
        …
        public void run() {
                while (true) {
                        pump.receive();
                        pump.send(flow.getValue())
                }
        }
}
```

Argument: …

```
class Pump extends Thread {
        …
        public void run() {
                error = false;
                while (!error) {
                        if (state==On) {
                                flow.send(); flow.receive(x);
                                if ((x =0) {
                                        error = true;
                                        pump.setOff()
                                }
                        }

                }
        }
}
```

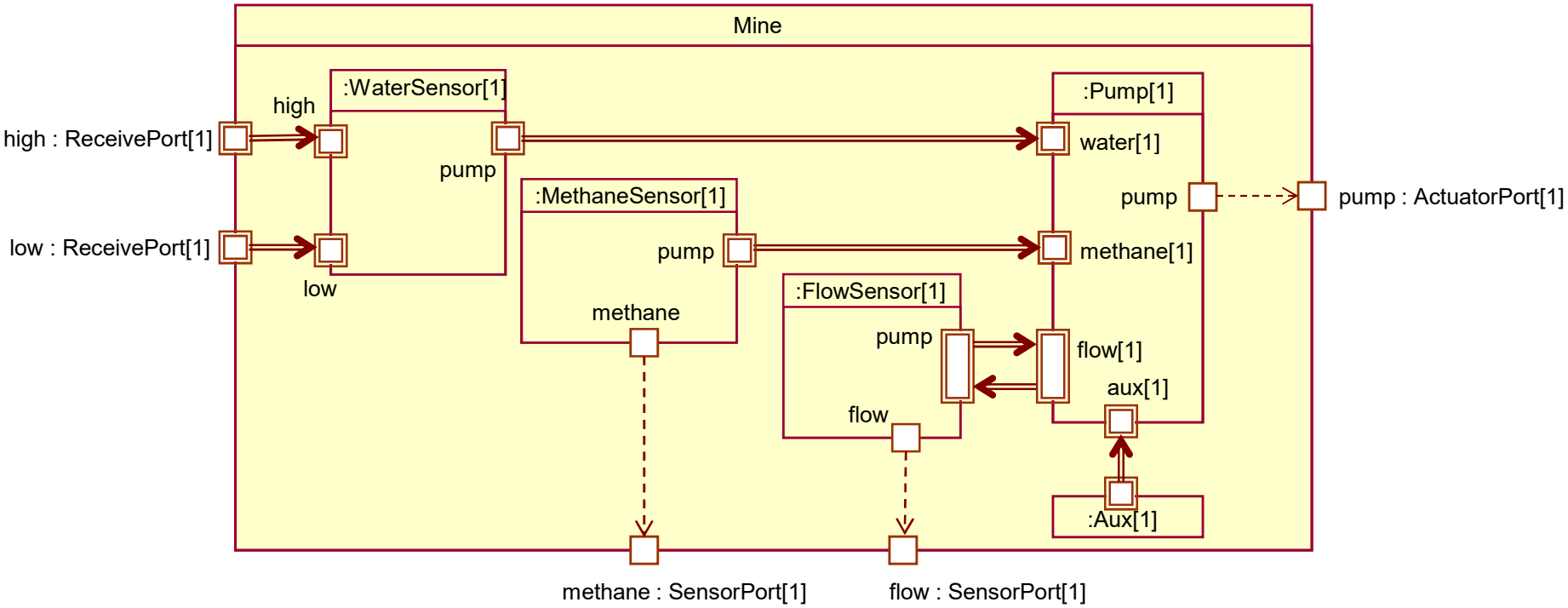
… Sequential Cohesion

```
class Aux extends Thread {
        …
        public void run() {
                while (true) {
                        pump.send();
                }
        }
}
```

… Sequential Cohesion

```
class Pump extends Thread {
        …
        public void run() {
                error = false;
                while (!error) {
                        aux.receive();
                            if (state==On) {
                                        flow.send(); flow.receive(x);
                                        if ((x =0) {
                                                error = true;
                                                pump.setOff()
                                        }
                            }
                }
        }
}
```

Refinement Step (3)



Final Version Pump

```
class Pump extends Thread {
        …
        public void run() {
                while (not error) {
                        port=receiveAll();
                        if (port=water)       high = (port.receive()==H);
                        if(port=methane)   safe = (port.receive()==0);
                        if(port=aux) {
                                port.receive();
                                if (state==On) { flow.send(); error = (flow.receive()==0);}
                        }

                        if(state=Off && high && safe && !error ) {
                                state=On; pump.setOn();
                        } else if(state=On && (!high || !safe || error)) {
                                state=Off; pump.setOff();
                        }
                }
        }
}
```

Execution (Scheduling) Directives

- The program can not be executed straightforwardly
  - The programming language doe not contain any reference to time
    - no wait(time) instruction
    - no indication of periods and deadlines
- These aspects pertain to the way the program is executed over time, not the program logic
- Periodic execution of instances of active classes MethaneSensor and Aux.
- Message (sporadic-event) driven execution of instance of active class Water.
- Also message driven execution of instances of active class Pump and FlowSensor.

- *What are the tasks for schedulability analysis??*

Task assignment

- Periodic task (time triggered)
  - Methane measurement
    - Body *MethaneSensor* and message handling *Pump*
  - Flow measurement
    - Body *Aux* and message handling *Pump* and *FlowSensor*
- Sporadic task (message triggered)
  - Water measurement
    - Body *WaterSensor* and message handling *Pump*
- Resources (these will cause blocking!!)
  - Shared by all tasks
    - Message handling of *Pump* and *FlowSensor*

Task parameters

tasks/resources	type	period	deadline	worst-case execution times
$\tau_{Methane}$	periodic	$T_{Methane}$	$dc - T_{Methane}$	$C_{Methane}$
τ_{Flow}	periodic	T_{Flow}	$df - T_{Flow}$	C_{Flow}
τ_{Water}	sporadic	dlh	$\min(dh,dl)$	C_{Water}
Pump	shared			$C_{Pump,Flow}$ $C_{Pump,Water}$ $C_{Pump,Methane}$
FlowSensor	shared			included in $C_{Pump,Flow}$

Schedulability Formula

- From earlier schedulability theory (see also Burns and Wellings, 1996)
  - Recursive relation for the response time of a task

$$r_i = \sum_{j=1}^{i-1} C_j \left\lceil \frac{r_i}{T_j} \right\rceil + C_i + \max_{j=i+1}^{m} C_{Mine,j}$$

Blocking

Task values

Task	C_i	$C_{Pump,i}$	*Di*	T_i	p_i
$\tau_{Methane}$	12u	6u	30 - $T_{Methane}$	$T_{Methane}$	1
τ_{Flow}	14u	13u	100 - T_{Flow}	T_{Flow}	2
τ_{Water}	9u	5u	10,000	100,000	3

- We are trying to find the slowest processor, i.e. the largest u (in milliseconds per statement), under which deadlines are still guaranteed.

Schedulability Analysis (DM)

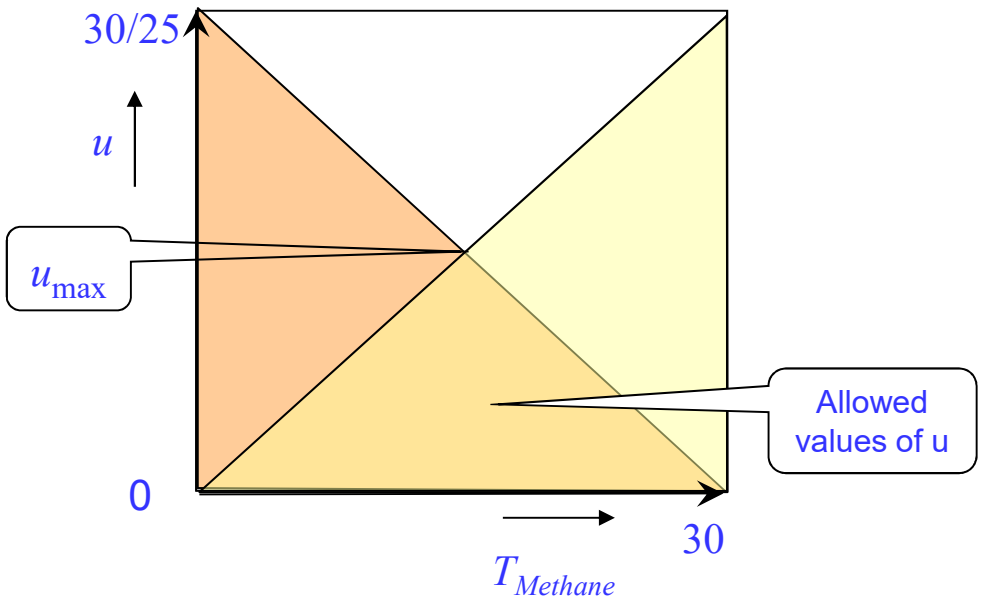
$$r_i = \sum_{j=1}^{i-1} C_j \left\lceil \frac{r_i}{T_j} \right\rceil + C_i + \max_{j=i+1}^{m} C_{Mine,j}$$

Blocking

– Consider task with shortest deadline ($\tau_{Methane}$) $t = 30 - T_{Methane}$

- $W_M(t) = C_1 + B_1 = 12u + 13u = 25u \leq 30 - T_{Methane}$ *(finish within deadline)*
  $\leq T_{Methane}$ *(finish within period)*

- ***Maximum value of u =0.6 when $T_{Methane}$=15 msec***



Schedulability Analysis (cont'd)

– Consider this u_{max}. Is it also ok for the other deadlines?

Check next shortest deadline task (τ_{Flow})

$t = 15, 30, 45, \ldots T_{Flow}$ (msec, order not certain)

- $W_C(15) = C_1 + C_2 + B_2 = 12u + 14u + 5u = 31u \leq 15$ *(finish within period)*
  $18.6 \leq 15$ **no**
- $W_C(30) = 2C_1 + C_2 + B_2 = 24u + 14u + 5u = 42u \leq 30$ *(finish within period)*
  $25.2 \leq 30$ **ok**

Also finish within deadline: $25.2 \leq 100 - T_{Flow}$

Maximum T_{Flow} = 74.8 msec *(gives lowest processor utilization)*

Check next task (τ_{Water})

$t = 15, 30, 45, 60, 74.8, 75, 90 \ldots 100{,}000$ (msec)

- $W_W(15) = C_1 + C_2 + C_3 = 12u + 14u + 9u = 35u \leq 15$ **no**
- $W_W(30) = 2C_1 + C_2 + C_3 = 24u + 14u + 9u = 47u \leq 30$ **ok**

Final Version Pump

```
class Pump extends Thread {
        private high=false, safe=true, error=false, state=Off
        public synchronized void setWater(Value value) {
                high = (value==H);
                generateResponse();
        }
        public synchronized void setMethane(Value value) {
                safe = (value==0);
                generateResponse();
        }
        public synchronized void run() {
                while (waitForNextPeriod(Tf) {
                                if (state==On) error = (flow.getValue()==0);
                                generateResponse();
                }
         }
         private void generateResponse() {
                if(state=Off && high && safe && !error ) { state=On; }
                else if(state=On && (!high || !safe || error)) { state=Off; }
         }
}
```